Amendments to the Claims

1. (*Currently Amended*) A method for partitioning a specification in a source code ~~(301)~~;

characterized in that the method comprises the following steps:

converting the specification into a plurality of abstract syntax trees ~~(101)~~; partitioning the plurality of abstract syntax trees ~~(101)~~ into at least a first set ~~(201)~~ and a second set ~~(203)~~, the first set of abstract syntax trees ~~(201)~~ to be implemented by a first processor ~~(GP, COPA)~~ and the second set of abstract syntax trees ~~(203)~~ to be implemented by a second processor ~~(COP, COPB)~~.

2. (*Original*) A method for partitioning a specification in a source code according to Claim 1, wherein the second processor is a co-processor.

3. (*Original*) A method for partitioning a specification in a source code according to Claim 2, wherein the first processor is a general-purpose processor.

4. (*Currently Amended*) A method for partitioning a specification in a source code according to Claim 1 further comprising the following step:

converting the first set of abstract syntax trees ~~(201)~~ to a first partial specification in the source code ~~(309)~~ and converting the second set of abstract syntax trees ~~(203)~~ to a second partial specification in the source code ~~(311)~~.

5. (*Currently Amended*) A method for partitioning a specification in a source code according to Claim 1 wherein the step of partitioning the plurality of abstract syntax trees ~~(101)~~ into a first set of abstract syntax trees ~~(201)~~ and a second set of abstract syntax trees ~~(203)~~ comprises a step of out-lining at least one abstract syntax tree based on profile data.

6. (*Currently Amended*) A method for partitioning a specification in a source code according to Claim 1 wherein the step of partitioning the plurality of abstract syntax trees ~~(101)~~ into a first set of abstract syntax trees ~~(201)~~ and a second set of abstract syntax trees

~~(203)~~ comprises a step of out-lining at least one abstract syntax tree based on programmer provided information.

7. (*Currently Amended*) A co-design method for producing a target system ~~(601, 701)~~, wherein the target system comprises a first processor ~~(GP, COPA)~~ and at least a second processor ~~(COP, COPB)~~;

the co-design method comprising the method for partitioning a specification in a source code according to Claim 1.

8. (*Currently Amended*) A co-design method for producing a target system according to Claim 7, wherein the method for partitioning a specification in a source code further comprises the following step:

converting the first set of abstract syntax trees ~~(201)~~ to a first partial specification in the source code ~~(309)~~ and converting the second set of abstract syntax trees ~~(203)~~ to a second partial specification in the source code ~~(311)~~.

9. (*Currently Amended*) A co-design method for producing a target system according to Claim 8 wherein the second processor is a co-processor and wherein the second partial specification ~~(311)~~ is converted to a specification of the co-processor ~~(319)~~.

10. (*Currently Amended*) A co-design method for producing a target system according to Claim 9, wherein the first processor is a general-purpose processor and wherein the first partial specification ~~(309)~~ is converted to object code ~~(315)~~ by means of a compiler

11. (*Currently Amended*) A co-design method for producing a target system according to Claim 10, further comprising a step for defining an interface between the general-purpose processor ~~(GP)~~ and the co-processor ~~(COP)~~.

12. (*Original*) A co-design method according to Claim 9, wherein the specification of the co-processor comprises a specification of an ASIC.

13. (*Original*) A co-design method according to Claim 9, wherein the specification of the co-processor comprises a specification of a programmable processor.

14. (*Original*) A co-design method according to Claim 9, wherein the specification of the co-processor comprises a specification of a reconfigurable processor.

15. (*Original*) A co-design method according to Claim 11, wherein the interface between the general-purpose processor (GP) and the co-processor (COP) comprises a remote function call;

the remote function call having a set of parameters;

the set of parameters comprising an identifier for the function to be called, at least one reference pointing to the input data of the function to be called and at least one reference pointing to the result data of the function to be called.

16. (*Original*) A co-design method according to Claim 15 wherein the set of parameters of the remote function call further comprises a reference to a memory location used for storing information on the return status of the function to be called.

17. (*Currently Amended*) A co-design method according to Claim 7 wherein the target system ~~(601, 701)~~ further comprises a system memory (SM) and a system bus (SB);

the system memory (SM), the first processor and the second processor being coupled by the system bus (SB).

18. (*Original*) A co-design method according to Claim 10 wherein the general-purpose processor (GP) is a digital signal processor.

19. (*Original*) A partitioning compiler program product, wherein the partitioning compiler program product is arranged for implementing all the steps of the method for partitioning a specification in a source code according to Claim 1, when said partitioning compiler program is run on a computer system.